Understanding Kubernetes - Container Orchestration technology

KUBERNETES, ALSO KNOWN AS K8'S, IS AN OPEN SOURCE SYSTEM FOR AUTOMATING DEPLOYMENT, SCALING AND MANAGEMENT OF CONTAINERISED APPLICATIONS.

Some of the Key advantages of Kubernetes are

1. HA of containerised application
2. Load balancing
3. Scale up and down based on the traffic

<u>**Components**</u> - that are installed when we setup Kubernetes

- API server:
  - Frontent for kunernetes
  - CLI, users communicate with API server to connect to the Kubernetes cluster
- ETCD:
  - key-value store
  - Store all the data required to manage the cluster
- Scheduler :
  - Distributes containers to nodes
  - Watches for newly created containers and assign them to a particular node
- Controller:
  - Brain behind the orchestration
  - Its main responsibility is to react and action when Node, container or endpoint goes down
- Container Runtime:
  - underlying software to run container. For example Docker
- Kubelet:
  - Agent that runs on each node in cluster
  - Ensure that containers are running as expected.
- Kubectl:
  - CLI for kubernetes

Some Basics Kubernetes Concepts

- Node:
  - Machine, Physical or Virtual
  - Also known as Minions
  - It is a machine where Kubernetes will launch containers
  - Basically we have 2 types of Nodes. Master and Worker
- Worker Node
  - Worker Nodes are the ones which has Kubelet agent installed on them.
  - This will have container runtime installed for example Docker.
- Master:
  - Does the actual Orchestration of containers on worker nodes.
  - This will not have any container running on it. Hence no need of Container runtime
  - This Node will have kube-apiserver component
  - This will also have controller and scheduler and etcd components of K8's
- Cluster:
  - Comprises of more than one nodes
  - Helps to achieve HA, Load balancing

Basics Cmds

- To run the pod

```
kubectl run my-app —image=my-app
```

- to get the pods

```
kubectl get pods
```

- to get the pod where the node is running

```
kubectl get pod -o wide
```

-to get the pod with additional details like Ip address, port, Host port, events etc

```
kubectl describe pod my-app
```

-to delete pod

```
kubectl delete pod my-app
```

YAML in Kubernetes - used to define Object Configuration file

YAML is data serialization standard for all programming languages. And in kubernetes we use it to define the objects.

YAML structure for creating any kind (Objects)

```
*****************************************************

apiVersion:

kind:

metadata:

spec:

*****************************************************
```

More details on the 4 key sections:

apiVersion: This is a string field and it takes the predefined value based on the 'kind'or object it will create. Please refer to the below kind-apiVersion Mapping to determine the value in configuration file of the 'kind' you intend to create

Kind-apiVersion Mapping

Kind	Version
POD	v1
Service	v1
Replication Controller	v1
ReplicaSet	apps/v1
Deployment	apps/v1

kind: it is an object you intend to create. You may refer to above kind-apiversion Mapping table to understand the different objects that you can create in cybernates ecosystem.

Metadata: Is the metadata of the object like it can have name or labels etc.

Example: below can be the metadata for the httpd container

```
metadata:
    name: Httpd
    labels:
        tier: webServer
```

Another advantage of having Metadata is that using the labels section, we can group the multiple container/pods.

spec: this hold the actual information of the object which Kubernetes will create like container name and container image name, evironment variables used by the objects etc.

Get Started with minikube - your local Kubernetes on your personal machine

minikube is local Kubernetes, focusing on making it easy to learn and develop for Kubernetes.
All you need is Docker (or similarly compatible) container or a Virtual Machine environment, and Kubernetes is a single command away: `minikube start`

```
For now just to get your hands dirty with Kubernetes you may go
ahead an setup a minikuube. Later in the course we will look at
multi Node cluster using  kubeadm
```

Install MiniKube: https://kubernetes.io/docs/tasks/tools/install-minikube/

PODs - Single instance of application.

- POD
  - And is the smallest object that you can create in Kubernetes.
  - Can have more than one containerised application. For example httpd
  - Multiple containers within POD (or Multiple containers PODs) shouldn't be of same kind.
  - You can create pods from the command line using the below command:

```
kubectl run httpd —image=httpd
```

here the image is the docker image hosted on docker hub

Lets create pod with YAML

1. First create an object configuration file using YAML as shown below. Can run the below cmd as it is on machine running on mac or linux

```
cat > pod_definition.yml
    apiVersion: v1
    kind: Pod
    metadata:
            name: httpdServer
            labels:
                    tier: webServer
    spec:
            containers:
                    - name: httpd
                            image: httpd
```

2. Then you can use create or apply cmd to create the pod using the above yaml file. **create** will create an object. In this case it is pod. **apply** usually used to apply the modified yaml changes to the created object but in case the object is not created apply will create a new objects.

```
kubectl create -f pod_definition.yml or
kubectl apply -f pod_definition.yml
```

- To get the pods

```
kubectl get pods
```

- To Edit the pod at runtime. Example if you want to change the image name with specific version then run below cmd.

```
kubectl edit pod httpdServer
```

- If you want to create he yaml configuration file using the kubectl cmd then run the below

```
kubectl run httpdServer—image=httpd —dry-run=client -o yaml >
pod._definition.yml
```

Replication Controllers:

- Ensure the HA or it ensure the expected number of POD running in cluster. This is achieved using "replicas" section in yml template file
- Load Balancing & Scaling
- Is now Replaced by Replica set. It is recommended by Kubernetes
- Only main difference between the replica set and replication controller is that replica set allows to manage the pods that were created outside the replica set definition yaml meaning pods created before the replica set creation or after. It is possible with "selector" section.

cmd examples to create replication controllers using yaml object definition file:

1. First create an object configuration file using YAML as shown below. Can run the below cmd as it is on machine running on mac or linux to create 3 application instances. configuration is in "**replicas**" section

```
cat > replicationController.yaml
apiVersion: v1
kind: ReplicationController
metadata:
        name: myapp-rc
        labels:
                app: myapp
                type: web-server
spec:
        template:
                metadata:
                name: httpdServer
                labels:
                        app: httpd
                        type: web-server
                spec:
                Containers:
                        - name: httpd
                        image: httpd
        replicas: 3
```

2. Run below cmd to create replicationController

```
kubectl create -f replicationController.yaml
```

3. To get the replication controller

```
kubectl get replicationController
```

cmd examples to create replicaSet using yaml object definition file:

1. First create an object configuration file using YAML as shown below. Can run the below cmd as it is on machine running on mac or linux to create 2 application instances. configuration is in "**replicas**" section and grouping of pods is done by the "selector" section

```
cat > replicaSet.yaml
apiVersion: v1
kind: ReplicaSet
metadata:
        name: myapp-replicaSet
        labels:
                app: myapp
                type: web-server
spec:
        template:
                metadata:
                name: httpdServer
                labels:
                        app: httpd
                        type: web-server
                spec:
                Containers:
                        - name: httpd
                        image: httpd
        replicas: 3
        selector:
                matchLabels:
                app: httpd
```

2. Run below cmd to create replicaSet

```
kubectl create -f replicationSet.yaml
```

3. To get the replication controller

```
kubectl get replicasets.apps
```

Deployments

- Deployments internally creates replicaset

Lets create a Deployment using YAML template file

1. Run the below cmd as it is on machine running on mac or linux to

```
cat > deployment.yaml
    apiVersion: apps/v1
    kind: Deployment
    metadata:
            name: frontend
            labels:
            app: mywebsite
            tier: frontend
    spec:
            replicas: 4
            template:
                    metadata:
                    name: myapp-deployment
                    labels:
                            app: httpd
            spec:
                    containers:
                            - name: httpd
                            image: httpd
            selector:
                    matchLabels:
                            app: httpd
```

2. Run the below cmd to create deployment

```
kubectl create -f deployment.yaml
```

Few more cmds To play with

- To scale the deployment

```
kubectl scale deployment —replicas=3 myapp-deployment
```

- Deployment rollout cmd

```
kubectl rollout status deployment/myapp-deployment
```

- Deployment rollout history

```
kubectl rollout history deployment/myapp-deployment
```

<u>Deployment Strategy</u> - Two Types

- Recreate
    - Destroy first then creates new later.
    - Application downtime
- Rolling update
    - Destroy one application instances at a time.
    - No application downtime
    - Default strategy
    - Creates a new replicaSet under hood for easy rollback

<u>kubectl continues</u>

Kubectl apply cmd can be used to apply any changes to the kind/Objects

For that we may first need to update the yaml file and then run the 'apply' cmd to apply the changes.

If you want to change the image then another way of doing the update to image is by running the below cmd. But Please note, below cmd won't update the yaml file so make sure in future while creating the kind the image is update to appropriate version otherwise the old changes will be applied to new instances.

```
kubectl set image deployment myapp-deployment httpd=httpd-1.2
```

- Rollback cmd

```
kubectl rollout undo deployment/myapp-deployment
```